

-1-

Date: <u>6/21/01</u>	Express Mail Label No. <u>EL551752379US</u>
----------------------	---

Inventor(s): David A. Brown

Attorney's Docket No.: 2037.2017-000

## METHOD AND APPARATUS FOR PHYSICAL WIDTH EXPANSION OF A LONGEST PREFIX MATCH LOOKUP TABLE

### RELATED APPLICATION(S)

This application claims the benefit of U.S. Provisional Application Nos.  
5 60/212,966 filed on June 21, 2000, 60/258,436 filed on December 27, 2000, and  
60/294,387 filed on May 30, 2001. The entire teachings of the above applications are  
incorporated herein by reference.

### BACKGROUND OF THE INVENTION

The Internet is a set of networks connected by routers. A router maintains a  
10 routing table that indicates for each possible destination network, the next hop to which  
a received data packet should be forwarded. The next hop may be another router or the  
final destination.

An Internet Protocol ("IP") data packet received at a port in a router includes an  
IP destination address. The IP destination address is the final destination of the IP data  
15 packet. Currently there are two versions of IP, IP version 4 ("IPv4") and IP version 6  
("IPv6"). IPv4 provides a 32-bit field in an IP header included in the data packet for  
storing the IP destination address. The router forwards a received data packet to a next-  
loop router or the final destination if the destination is the local network, dependent on  
the IP destination address stored in the IP header.

20 A 32-bit IPv4 destination address provides 4 billion possible routes or  
destinations. An Internet router typically stores a next hop for 50,000 of the 4 billion

possible destinations. However, the number of stored routes will increase with the growth of the Internet and the widespread use of IPv6.

Originally, the IP address space was divided into three classes of IP addresses; A, B and C. Each IP address space was divided into a network address and a host address. Class A allowed for 126 networks and 16 million hosts per network. Class B allowed for 16382 networks with 64,000 hosts per network and class C allowed for 2 million networks with 256 hosts per network. However, dividing the IP address space into different classes reduced the number of available IP addresses. Class C only allowed a maximum of 256 hosts per network which is too small for most organizations. Therefore, most organizations were assigned a Class B address, taking up 64,000 host addresses which could not be used by other organizations even if they were not used by the organization to which they were assigned. Hosts in an organization with a Class B IP address all use the same network address in the 16 Most Significant Bits ("MSBs"), for example, 128.32.xx.xx.

Classless InterDomain Routing ("CIDR") was introduced to free up unused IP host addresses. The remaining unused networks are allocated to organization in variable sized blocks. An organization requiring 500 addresses gets 500 continuous addresses. For example, an organization can be assigned 500 available addresses starting at 128.32.xx. The number of routes stored by a router has increased since the introduction of Classless InterDomain Routing. Classless InterDomain Routing requires longest prefix matching to find the corresponding route instead of searching for a matching network address in order to find the corresponding next hop for the IP destination address. For example, a search can no longer stop after the 16 Most Significant Bits ("MSBs") of a Class B IP address, for example, 128.32.xx because 128.32.4.xx may be assigned to another organization requiring a different next hop.

One method for searching for a longest prefix match for a key is through the use of a binary tree search. A binary tree search matches a 32-bit input bit by bit down to 32 levels, requiring 32 searches to finding the entry matching the 32-bit key. Another method for searching for a match is through the use of a Patricia tree. A Patricia tree

reduces the number of searches required if there are no entries down a leaf of the binary tree.

Yet another method for efficiently searching for a next hop associated with an IP destination address is described in PCT application Serial Number PCT/SE98/00854  
 5 entitled "Method and System for Fast Routing Lookups" by Brodnick *et al.* filed on May 11, 1998. The method described by Brodnick reduces the number of next hops stored by not storing duplicate routes. By reducing the number of next hops, the memory requirement is reduced so that a route lookup table can be stored in fast cache memory.

10 Brodnick *et al.* divides the binary tree into 3-levels. Dividing the binary tree into 3-levels reduces the number of searches to three. The indexed entry in the first level indicates whether the search can end at the first level with the route taken from the entry, or the search must continue to a subsequent level using a further portion of the IP destination address.

15 Fig. 1A illustrates a prior art 64K (65536) bit map representing the first level of a binary tree. A 64K bit map 30 represents the leaves or nodes 44 of the binary tree at depth 16, with one bit per node 44. The bit map is divided into bit-masks of length 16. There are  $2^{12} = 4096$  bit masks in the 64k bit map. One bit mask is shown in Fig. 1A. A bit in the bit map 30 is set to '1' if there is a subtree or a route index stored in an array  
 20 of pointers corresponding to the node 44. A bit in the bit map 30 is set to '0' if the node shares a route entry with a previous node 44.

Fig. 1B illustrates prior art lookup table implemented in cache memory. The lookup table includes an array of code words 36, an array of base indexes 34 and a map table 40. A 32-bit IP address 38 is also shown in Fig. 1B. A codeword 46 is stored in  
 25 the array of code words 36 for each bit mask in the bit map 30 (Fig. 1A). The code word 46 includes a six-bit value 46a and a 10-bit offset 46b. A base index 42 is stored in the array of base indexes 34 for every four code words 46 in the array of code words 36.

The array of code words 36, array of base indexes 34 and map table 40 are used to select a pointer in an array of pointers (not shown). The pointer stores a route index or an index to perform a further search.

A group of pointers in the array of pointers is selected by selecting a code word 5 46 in the array of code words 36 and a base index 42 in the array of base indexes 34. The code word 46 is selected using the first 12 bits 50 of the IP address 38. The base index 42 is selected using the first 10 bits 48 of the IP address 38. The correct pointer in the group of pointers is selected using the map table 32.

The 10-bit value 46b in the selected code word 36 is an index into the map table 10 32. The map table 32 maps bit numbers within a bit-mask to 4-bit offsets. The offset specifies the pointer within the selected group of pointers in the array of pointers. The 10-bit value 46b selects the row in the map table 32 and bits 19:16 of the IP address 52 selects the 4-bit offset 54.

Thus, a search for a pointer requires the following cache memory accesses: (1) 15 read a 16 bit code word 46; (2) read a 16-bit base address 42; (3) read a 4 bit offset 54 from the map table 32; (4) read a pointer at a pointer index where the pointer index is the sum of the base address 42, the code word offset 46a and the 4-bit offset 54.

The same memory accesses are required for each level of the binary tree. Thus, a search of three levels for a 32-bit IPv4 address requires 12 memory accesses. As 20 many as forty-eight memory accesses can be required to perform a longest prefix search for a 128-bit IPv6 address.

#### SUMMARY OF THE INVENTION

U.S. Patent Application Serial No. 09/733,627 entitled "Method and Apparatus for Longest Match Address Lookup," filed December 8, 2000 by David A. Brown 25 describes a lookup unit for performing multiple level searches with portions of a search key in successive mappers, entries in the mappers outputting route indexes or providing partial indexes to subsequent mappers. The length of the search key is limited by the number of search levels in the lookup units.

In accordance with the invention, a longest prefix match lookup matrix allows searching from longer search keys including search keys of different lengths such as the 32-bit IPv4 and 128 IPv6 addresses. A lookup matrix includes a master lookup unit and at least one non-master lookup unit. The master lookup unit and the non-master lookup unit include a plurality of mappers. The mappers in the master lookup unit are indexed by portions of a first portion of a search key to output a route index for the search key or partial indexes to subsequent mappers. The mappers in the non-master lookup unit are indexed by portions of a next portion of the search key and a partial index from a prior lookup unit to output the route index for the search key or another partial index to a subsequent non-master lookup unit.

The route index corresponding to the search key is stored in a single location in one of the lookup units. The length of the search key is variable and may be expanded by adding an additional non-master lookup unit. The search key may include a 32-bit IPv4 address or a 128 IPv6 address. If the search key includes a 32-bit IPv4 address, the route index corresponding to the search key is found after a first search of the plurality of mappers. The partial index may be a subtree index.

#### BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

- Fig. 1A illustrates a prior art bit map representing the first level of a binary tree;
- Fig. 1B illustrates a prior art lookup unit implemented in cache memory;
- Fig. 2 illustrates a forwarding engine coupled to a lookup unit matrix according to the principles of the present invention;

Fig 3A illustrates a 64-level binary tree representation of entries stored in the lookup unit matrix shown in Fig. 2;

Fig. 3B illustrates the master lookup unit in the lookup unit matrix shown in Fig. 2;

5 Fig. 4 illustrates the types of mapper entries which can be stored in any of the mappers in the lookup unit shown in Fig. 3B;

Fig. 5 illustrates one of the indirect mappers in the lookup unit shown in Fig. 3B; and

10 Fig. 6 is a flowchart illustrating the steps for searching for a route stored in a location .

## DETAILED DESCRIPTION OF THE INVENTION

A description of preferred embodiments of the invention follows.

15 Fig. 2 illustrates a forwarding engine 108 coupled to a longest prefix match lookup unit matrix 100 according to the principles of the present invention. The lookup unit matrix 100 includes a plurality of lookup units 150a, 150b. The lookup unit matrix 100 performs a multi-level search in one or more of the lookup units 150a, 150b for a value corresponding to a portion of the search key 104 forwarded as mapper keys 104a, 104b by the forwarding engine 108. A route index 102 for a search key 104 which may  
20 be longer than the lookup unit's mapper key is stored in a mapper entry in one of the lookup units 150a, 150b.

In one embodiment, the search key 104 is an Internet Protocol ("IP") address. The forwarding engine 108 and the lookup unit matrix 100 provide a route index 102 to a next hop or destination corresponding to the IP address. Well known standard IP  
25 addresses include the 32-bit IPv4 address and the 128-bit IPv6 address. A 40-bit mapper key 104a can include a 32-bit IPv4 address and an 8-bit prefix. The invention is described for an embodiment including two lookup units and a 64-bit search key. The search key can be expanded to a 128-bit IPv6 address by adding more lookup units. In the embodiment shown, a 64-bit search key longer than a 32-bit IPv4 address is input to

the forwarding engine 108. The forwarding engine 108 divides the 64-bit search key into mapper keys 104a and 104b and forwards mapper keys 104a and 104b to the lookup unit matrix 100.

The lookup unit matrix 100 performs a multi-level search in master lookup unit 150a for a route index 102 corresponding to the first portion of the search key 104 forwarded as 40-bit mapper key 104a. If the search of master lookup unit 150a does not result in a route index 102 corresponding to the first 40-bits of the search key 104a, a subsequent search is performed in non-master lookup unit 150b for a value corresponding to the next 24-bits of the key 104b forwarded by the forwarding engine 108 and the search result 106 from master lookup unit 150a.

In the embodiment shown, the search key 104 is 64-bits long and the lookup unit matrix 100 includes two lookup units 150a, 150b. Master lookup unit 150a performs a search for a route index corresponding to the first 40-bits of the 64-bit search key 104. Non-master lookup unit 150b performs a search for a route index corresponding to the next 24-bits of the 64-bit search key 104 and the result of the search of master lookup unit 150a. The search key 104 can be expanded further by adding more non-master lookup units 150b to the lookup unit matrix 100. Each additional non-master lookup unit 150b expands the search key 104 by 24-bits.

The lookup unit matrix 100 can store route indexes for both IPv4 and IPv6 addresses. A single search cycle for the 32-bit IPv4 address results in a route index 102 corresponding the longest prefix match for the 32-bit IPv4 address stored in master lookup unit 150a in the lookup unit matrix 100. The resulting route index 102 is forwarded to the forwarding engine 108.

A 128-bit IPv6 address is longer than the 40-bit mapper key 104a. Thus, a search of master lookup unit 150a may not be sufficient. Typically, there are more route indexes for IPv4 addresses stored in a router than for IPv6 addresses. The length of the mapper key 104a is therefore selected such that a search for a route index 102 corresponding to an 8-bit prefix and a 32-bit IPv4 address can be performed in a single search of master lookup unit 150a. Thus, only infrequent searches for route indexes for

longer search keys, such as 128-bit IPv6 addresses require searching multiple lookup units 150a, 150b.

A search of a lookup unit 150a, 150b is described in co-pending patent application Attorney Docket Number 2037.2004-001 U.S. Patent Application Serial No. 09/733,627 filed on December 8, 2000 entitled "Method and Apparatus for Longest Match Address Lookup," by David A. Brown incorporated herein by reference in its entirety.

The search key is expanded by combining a plurality of lookup units. A single lookup unit stores routes for search keys less than or equal to the lookup unit's mapper key or a plurality of lookup units are combined to store routes corresponding to keys longer than the lookup unit's mapper key.

Each lookup unit 150a, 150b includes a device identifier 232 set at power up by pin straps. The state of the device identifier 232 determines whether the size of the mapper key for each lookup unit 150a, 150b is 40-bits or 24-bits. If the device identifier 232 identifies the lookup unit as the master lookup unit 150a, the mapper key 104a is 40-bits and the first level mapper search in the master lookup unit 150a starts with the 16 Most Significant Bits ("MSBs") of the search key 104. If the device identifier 232 identifies the lookup unit as a non-master lookup unit 150b, the mapper key 104b is the next 24-bits of the search key 104 and the second level mapper search in the non-master lookup unit 150b starts with the first 8-bits of mapper key 104b and the result of the search of lookup unit 150a.

In a search of the lookup unit matrix 100 for a route index 102 corresponding to the search key 104, the most significant 40-bits of the search key 104 are forwarded by the forwarding engine 108 to the lookup unit matrix 100 as mapper key 104a together with a "search" command on the command bus 112. A search is performed in master lookup unit 150a and non-master lookup unit 150b. The search in master lookup unit 150a begins in the first mapper level because the device identifier indicates that master lookup unit is a master lookup unit. The search in non-master lookup unit 150b begins in the second mapper level irrespective of the command received because there are no



route indexes or partial indexes stored in the first level mapper in the non-master lookup unit 150b.

A lookup unit matrix 100 including eight lookup units 150 can store a route index 102 corresponding to a 208-bit key. The search is performed in the master  
 5 lookup unit 150a for a route index or partial index corresponding to the first 40-bits of the search key 104 which are forwarded as mapper key 104a to the master lookup unit 150a. Seven subsequent searches are performed, if necessary, dependent on the result of the search of the previous lookup unit and the next 24-bits of the search key 104 in the next seven lookup units in the lookup table matrix 100. The search in the other  
 10 seven lookup units begins in the second mapper level because the state of the device identifier 232 for each of the seven lookup units indicates that the lookup units are non-master lookup units.

In an alternate embodiment, lookup unit 150b can be configured to logically expand the second portion of the search key 104. For example, in the case of a 128-bit  
 15 search key the 40-bit mapper key can be forwarded to the master lookup unit 150a and the remaining 88 bits of the 128-bit search key can be searched repeatedly by non-master lookup unit 150b, 24-bits at a time with the first 24-bits searched with the previous result of the search of master lookup unit 150a. A method and apparatus for logically expanding a search key is described in co-pending patent application Attorney  
 20 Docket No. 2037-2018-000, filed on even date herewith, entitled "Method and Apparatus for Logically Expanding A Search Key", by David A. Brown incorporated herein by reference in its entirety.

Fig. 3A illustrates a 64-level binary tree representation of the entries stored in the lookup unit matrix 100 shown in Fig. 2. The 64-bit binary tree representation is  
 25 used to illustrate physical expansion of a lookup unit. A lookup unit is expanded by combining a plurality of lookup units 150a, 150b in a lookup unit matrix 100. The lookup unit matrix 100 stores a value corresponding to a search key 104 (Fig. 2) that is longer than the mapper key 104a, 104b (Fig. 2). In the embodiment shown, the search

key 104 (Fig. 2) is 64-bits long, mapper key 104a, is 40-bits long and mapper key 104b is 24-bits long; however, the invention is not limited to this configuration.

The 64-bit key can be represented as a 64-level binary tree. A search for an entry corresponding to a 64-bit key requires 64 searches to search bit by bit down to 64 levels. To reduce the number of searches, the 64 levels of the binary tree are divided into mapper levels 114a-g. Mapper level\_1 114a includes the first 16 of the 64 levels of the binary tree. However, for simplicity only 5 of the 16 levels are shown. Mapper level\_2 114b includes the next 8 levels of the 64-level binary tree, with three of the eight levels shown. Mapper level\_3 includes the next 8 levels of the 64-level binary tree, with three of the eight levels shown. Each of mapper levels\_4-7 also includes 8 levels of the 64-level binary tree with three of the eight levels shown. Master lookup unit 150a (Fig. 2) includes mapper levels\_1-4 and non-master lookup unit 150b, (Fig. 2) includes mapper-levels\_5-7. Each mapper level 114a-g includes a plurality of nodes. Dividing the 64-levels such that 16-levels (16 -MSBs) of the search key 104 (Fig. 2)) are in mapper level\_1 114a and 8-levels in mapper levels 114b-g appears to be optimal in the current memory technology; however, the invention is not limited to this configuration.

Fig. 3B illustrates the master lookup unit 150a in the lookup unit matrix 100 shown in Fig. 2. Non-master lookup unit 150b differs from master lookup unit 150a only in state of the device identifier 232. As shown, master lookup unit 150a includes four mappers 206a-d. The route index 102 for a search key 104 is stored in a route entry 302 in one of the mappers 206a-d. The mapper key 104a is 40-bits long allowing a search for a route entry corresponding to an 8-bit prefix and a 32-bit IPv4 address stored in one of the mappers 206a-d. The 8-bit prefix can be a Virtual Private Network (“VPN”) identifier associated with the 32-bit IPv4 address. For an IP address longer than the 32-bit IPv4 address, the 40-bit mapper key 104a includes the VPN and the 32 most significant bits of the IP address. Mapper 206a is a direct mapped mapper. Mappers 206b-d are indirect mappers and will be described later in conjunction with Fig. 5.

Direct mapped mapper 206a stores a route index 102 or a partial index for the L2 mapper 206b corresponding to the 16 MSBs of mapper key 104a. Thus, the L1 mapper 206a has  $2^{16}$  locations, one for each of the  $2^{16}$  nodes in the first mapper level 114a (Fig. 3A). The L1 mapper entry data 220a stored at the corresponding location in the L1 mapper 206a is forwarded to a pipeline 208 and to the L1 pointer selector 212. In the master lookup unit 150a, while command out 200 is set to "search", the L1 pointer selector 212 forwards the L1 mapper entry data 220a to the L2 mapper 206b dependent on the state of the device identifier 232. If the L1 mapper entry data 220a indicates that a search of the next level is required using the next eight bits of mapper key 110b, a search is performed in the L2 indirect mapper 206b dependent on the next eight bits of the mapper key 110b, and the L1 mapper entry data 220a forwarded by the L1 pointer selector 212.

The result of the second level search in L2 indirect mapper 206b is forwarded on L2 mapper entry data 220b to the pipeline 208 and to the L3 indirect mapper 206c. A third level search is performed in the L3 indirect mapper 206c dependent on the next eight bits of the mapper key 110c and the L2 mapper entry data 220b.

The result of the search of the L3 indirect mapper 206c is forwarded on L3 mapper entry data 220c to the pipeline 208 and to the L4 indirect mapper 206d. The L3 mapper entry data 220c determines if a fourth level search must be performed in the L4 indirect mapper 206d dependent on the last eight bits of the key 110d and the result of the search of the L3 indirect mapper 206c forwarded as L3 mapper entry data 220c.

The result of the fourth level search is provided on L4 mapper entry data 220d. If the route index 102 associated with the longest prefix match for search key 104 is stored in master lookup unit 150a, it is stored in only one location in one of the mappers 206a-d and forwarded to the pipeline 208. If the route index 102 is found in one of the mappers 206a-d, for example, mapper 206b a search of the remaining mappers 206c-d is not necessary and mappers 206c-d are not accessed. The pipeline 208 selects the route index 102 included in one of the mapper entry data 220a-d. For example, the

MSB of the mapper entry data 220a-d can provide an indication of whether a route index 102 is included.

By using a pipeline 208 in conjunction with the mappers 206a-d, multiple searches of a lookup unit 150a 150b with different values of mapper keys 104a can be performed in parallel. The pipeline 208 allows multiple searches of the lookup unit 150a, 150b to take place in parallel by storing the mapper entry data 220a-d for each mapper 206a-d associated with the 40-bit mapper key 104a until a search of each of the other mappers 206a-d has been completed, if required, to find route index corresponding to the 40-bit mapper key 104a.

Instead of performing 16 separate bit by bit searches for the first 16 bits of the search key 104 the mapper 206a is directly indexed with the first 16- MSBs of the search key 104. A search of mapper 206a in master lookup unit 150a is only performed for the first 16 bits of the search key 104. Mapper 206a in non-master lookup unit 150b is not used and thus it is not searched.

Returning to Fig. 3A, the nodes or leaves shown in mapper level<sub>1</sub> 114a include two routes 118, 116 labeled r0 and r1 respectively and two pointers to mapper level<sub>2</sub> 114b 130<sup>4</sup> and 130<sup>23</sup> labeled s0 and s1, respectively. A route index 102 for each route 118, 116 is stored in the L1 mapper 206a (Fig. 3B). Also, an address pointer for L2 mapper 206b (Fig. 3B) is stored in the L1 mapper 206a (Fig. 3B) for subtree index 130<sup>4</sup> and subtree index 130<sup>23</sup>. An address pointer stored for subtree index 130<sup>4</sup>, in L1 mapper 206a (Fig. 3B) indicates that a search of the next level is required in order to find a route index 102 associated with the mapper key 104a.

The value of any node in the tree can be determined by tracing a path from the root 118. Each node in the binary tree is shown with two children, a right child and a left child. The right child is chosen if the parent node is '1.' The left child is chosen if the parent node is '0'. Tracing the path from the root 118 to node 116, r1 is stored as the route index 102 in the L1 mapper 206a (Fig. 3B) for all keys with MSBs set to '010'. Tracing the path from the root node 118 to s0 node 130<sup>4</sup>, s0 is stored in the L1 mapper 206a (Fig. 3B) for all keys with MSBs set to '00011'.

The L1 mapper 206a (Fig. 3B) is a direct mapped mapper and stores a route index 102 for each bottom-level node or leaf in the bottom level of mapper level\_1 114a. The bottom level of mapper level\_1 114a is the sixteenth level of the 64-level binary tree. The sixteenth level has  $2^{16}$  nodes. However, for illustrative purposes level\_5 of the 64-level binary tree is shown as the bottom level of mapper level\_1 114a. The route indexes 102 shown in the L1 mapper 206a (Fig. 3B) correspond to level\_5  $130^1$ - $130^{32}$  nodes of mapper level\_1 114a. Tracing the path from the root node 118 to level\_5 nodes  $130^1$ ,  $130^2$ ,  $130^3$  the route index 102 is r0. Thus r0 is stored at index 00000, 00001, and 00010 in L1 mapper 206a. Node  $130^4$  stores a subtree index s0, thus s0 is stored in the L1 mapper 206a at index 00011. Similarly the route index 102 for level\_5 nodes  $130^5$ - $130^8$  is r0 thus locations at indexes 00100, 00101, 00110, and 00111 in the L1 mapper 206a (Fig. 3B) store r0. The route index 102 for level\_5 nodes  $130^9$ - $130^{12}$  is r1, thus locations at indexes 01000, 01001, 01010 and 01011 in the L1 mapper 206a (Fig. 3B) store r1.

Each location in the L1 mapper 206a (Fig. 3B) stores a route index 102 assigned to the level\_5 node  $300^1$ - $300^{32}$  directly or through a parent of the level-5 node  $300^{1-32}$  or an address pointer to the next mapper 206b (Fig. 3B). Mapper level\_4 114d includes two host nodes h0 at node 138 and h1 at node 140. A search for a host node requires a search of all bits of the search key 104. The route index 102 for h0 is stored in L4\_mapper 206d (Fig. 3B). Unlike the L1 mapper 206a (Fig. 3B), the L2 mapper 206b (Fig. 3B), L3 mapper 206c (Fig. 3B) and L4 mapper 206d (Fig. 3B) are not directly mapped.

Returning to Fig. 3B, seven search levels are required to search for the route index 102 corresponding to host node h2 in mapper level\_7 114g. However, each lookup unit 150a, 150b only includes four mappers 206a-d. Thus, if a search of nodes in levels 1-4 114a-d in mappers 206a-d in master lookup unit 150a does not result in a route index, a further search for a route entry or subtree entry for mapper level\_5 114e (Fig. 3A) stored in L2 mapper 206b in non-master lookup unit 150b (Fig. 2) is performed with the next portion of the search key 104 and the result of the search in

master lookup unit 150a. L1 pointer selector 212 determines whether the search of mapper 206b in non-master lookup unit 150b (Fig. 2) is being performed for a node in mapper level\_5 114e (Fig. 3A) or a node in mapper level\_2 114b (Fig. 3A) dependent on the device identifier 232 (Fig. 2).

5 If the search of mapper 206b in non-master lookup unit 150b (Fig. 2) is being performed for a node in mapper level\_5 114e (Fig. 3A), the search result 106 from the multi-level search of master lookup unit 150a (Fig. 2) is forwarded to the L2 indirect mapper 206b in non-master lookup unit 150b (Fig. 2). The forwarding engine 108 (Fig. 2) forwards the next 24-bits of the 64-bit search key 104 as mapper key 104b (Fig. 2) to  
10 non-master lookup unit 150b.

The search continues in the L2 indirect mapper 206b in non-master lookup unit 150b. The L1 pointer selector 212 in non-master lookup unit 150b forwards the search result 106 forwarded from master lookup unit 150a to the L2 indirect mapper 206b. The L2 indirect mapper 206b in non-master lookup unit 150b searches for an entry  
15 dependent on the search result 106 and the next 8-bits of the mapper key 104b.

Subtree indexes and route indexes for level\_5 114e (Fig. 3A) are stored in L2 indirect mapper 206b in non-master lookup unit 150b. Subtree indexes and route indexes for level\_6 114f (Fig. 3A) are stored in L3 indirect mapper 206c in non-master lookup unit 150b and subtree indexes and route indexes for level\_7 114g (Fig. 3A) are  
20 stored in L4 indirect mapper 206d in non-master lookup unit 150b.

Thus, the route index 102 for node labeled h2 in level\_7 114g (Fig. 3A) is stored in L4 indirect mapper 206d in non-master lookup unit 150b. The forwarding engine 108 (Fig. 2) issues one search request to the lookup unit matrix 100. The lookup unit matrix 100 provides the route index 102 corresponding to host node h2 after a multi-  
25 level search of lookup units 150a, 150b.

Fig. 4 illustrates the types of mapper entries which can be stored in any of the mappers 206a-d shown in Fig. 3B. A mapper entry for any node in the binary tree shown in Fig. 3A can store, a no-entry 300, a route entry 302 or a subtree entry descriptor 304. Each type of mapper entry 300, 302, 304 includes a subtree flag 306.

The state of the subtree flag 306 indicates whether the mapper entry is a subtree entry descriptor 304. If the subtree flag 306 is set to '1', the mapper entry is a subtree entry descriptor 304 and includes a L1 mapper entry data 220a (Fig. 3B). The L1 mapper entry data 220a (Fig. 3B) is the address of the subtree entry descriptor 304 stored in the next non-direct mapped mapper 206b-d (Fig. 3B). If the subtree flag 306 is '0', the no-entry flag 314 is checked to determine if the mapper entry is a no-entry 300 or a route entry 302. If the no-entry flag 314 is '0', the entry is a no-entry 300. If the no-entry flag 314 is '1', the entry is a route entry 302 and stores the route index 102 (Fig. 2) associated with the search key 104 (Fig. 2) in the route index field 310.

Fig 5 illustrates one of the indirect mappers 206b shown in Fig. 3B. Indirect mapper 206b in each of the lookup units 150a, 150b stores route entries 302 and subtree entry descriptors 304 corresponding to the nodes in mapper level\_2 114b (Fig. 3A) and mapper level\_5 114e (Fig. 3A). Mapper 206b includes a subtree memory 500, an index generator 504, a pointer selector 506 and a subtree mapper 502. In master lookup unit 150a (Fig. 2), the L1 mapper entry data 220a selected by the first portion of the mapper key 104a stored in mapper 206a (Fig. 3B) stores a partial index. The partial index is forwarded as the subtree memory index 230 to the subtree memory mapper 500. In non-master lookup unit 150b (Fig. 2), the search result 106 (Fig. 3B) from the result of searching master lookup unit 150a (Fig. 2) is forwarded as the subtree memory index 230. The subtree memory 500 includes a subtree entry 404 indexed by the subtree memory index 230. The subtree entry 404 includes a data field 406 and a pointers field 408.

Returning to Fig. 3A, the subtree entry 404 corresponds to the bottom level of one of the subtrees shown in mapper levels 114b-g. For example, if mapper level\_2 114b has eight levels, the bottom level of each subtree (not shown) has a maximum of  $2^8$  (256) routes, one for each of the  $2^8$  (256) nodes.

Continuing with Fig. 5, the subtree entry 404 provides access to 256 possible mapper entries corresponding to the 256 nodes on the bottom level of the subtree. The mapper entries are stored in the subtree mapper 502. To provide access to 256 possible

mapper entries, a dense subtree descriptor is stored in the data field 406. The data field 406 is 256 bits wide, providing one bit for each node at the bottom level of the subtree. A bit in the data field 406 is set to '0' if the mapper entry for the previous node is to be used and set to '1' to increment to the next mapper entry address if the next mapper entry stored in the subtree mapper 502 is to be used.

The pointers field 408 is 256 bits wide to allow for the storage of sixteen 16-bit pointers per logical row, with each pointer storing the base address for 16 contiguous mapper entries in the subtree mapper 502, to provide 256 mapper entries per logical row. Thus, the pointers field 408 can indirectly provide a pointer to a mapper entry in the subtree mapper 502 for each node in the bottom level of the subtree. The data field 406 and pointers field 418 are described in co-pending application U.S. Patent Application Serial No. 09/733,627 entitled "Method and Apparatus for Longest Match Address Lookup," filed December 8, 2000 by David A. Brown incorporated herein by reference in its entirety.

The subtree data stored in the dense subtree descriptor in the data field 406 and the subtree pointer stored in the pointers field 408 for a selected node in the subtree are forwarded to the index generator 504. The index generator 504 also receives the next eight bits of the mapper key 110b.

The index generator 504 generates the mapper address 512 of the mapper entry associated with the node in the bottom level of the subtree dependent on the next eight bits of the mapper key 110b, and the subtree entry 510 associated with the subtree. The subtree entry 510 includes the subtree data field 406 and subtree pointers field 408 storing subtree data and subtree pointers for the subtree selected by the subtree memory index 230. The mapper address 512 indexes the mapper entry in the subtree mapper 502. The subtree mapper 502 includes the same types of mapper entries as described in conjunction with Fig. 4 for the direct mapped mapper 206a. The contents of L2 mapper entry data 220b determine whether a subsequent search of the next mapper in the lookup unit 150a, 150b (Fig. 2) is required. A subsequent search is required, if the L2 mapper entry data 220b includes a subtree entry 304, indicating that there is another



subtree index 312 stored in a mapper entry in the subtree mapper 502 for the next mapper level 114c (Fig. 3A).

The next eight bits of the mapper key 110b select the node in the bottom level of the selected subtree. The subtree pointers 408 select a base address associated with the node in the subtree and the subtree data 406 selects the offset within the block of mapper entries associated with the base address.

The pointer generator 506 generates the L2 mapper entry data 220b to be forwarded to the L3 indirect mapper 206c (Fig. 3B) dependent on the L2 mapper output 508 from the subtree mapper 502, the L1 result 514 from the L1 direct mapped mapper 206a and the L2 index 518 received from the L2 index generator 504. The pointer generator 506 also generates the L2 result 516 to be forwarded to the L3 pointer generator in the next indirect mapper 206c (Fig. 3B).

Fig. 6 is a flowchart illustrating the steps for searching for a route index 102 (Fig. 2) corresponding to a search key 104 (Fig. 2) longer than the mapper key 104a, 104b for a lookup unit in the lookup unit matrix 100 shown in Fig. 3B. Fig. 6 is described in conjunction with Fig. 2 and Fig. 3B.

At step 600, the lookup units 150a, 150b in the lookup matrix unit 100 (Fig. 2) wait to receive a portion of a search key 104 (Fig. 2) on mapper key 104a, 104b. If a portion of search key 104 (Fig. 2) is received, processing continues with step 602.

At step 602, the lookup unit 150a, 150b (Fig. 2) examines the state of the device identifier 232 (Fig. 3B). If the device identifier 232 (Fig. 3B) is "master" and the command is "search", the search for a route index corresponding to mapper key 104a begins in L1 direct mapped mapper 206a in master lookup unit 150a (Fig. 2) and processing continues with step 604. If the device identifier 232 (Fig. 3B) is "non-master", the search begins in L2 indirect mapper 206b (Fig. 2) in non-master lookup unit 150b (Fig. 2) for a route index corresponding to mapper key 104b and the search result 106 from master lookup unit 150a and processing continues with step 614.

At step 604, master lookup unit 150a in lookup unit matrix 100 (Fig. 2) performs a search in the direct mapper 206a (Fig. 3B) for the 16-MSBs of the search key 104 (Fig. 2) forwarded as mapper key 104a. Processing continues with step 606.

At step 606, the next indirect mapper 206b-d in master lookup unit 150a or the  
5 next indirect mapper 206c-d in non-master lookup unit 150b in the lookup unit matrix 100 (Fig. 2) examines the result of the previous mapper search. If the result is a route index corresponding to the search key 104 (Fig. 2), processing continues with step 612. If not, processing continues with step 608.

At step 608, the search of the previous mappers 206a-d (Fig. 3B) in master  
10 lookup unit 150a (Fig. 3B) or mappers 206b-d in non-master lookup unit 150b did not result in a route index. The master lookup unit 150a (Fig. 3B) determines if there is another mapper 206b-d (Fig. 3B) or the non-master lookup unit 150b (Fig. 3B) determines if there is another mapper 206c-d (Fig. 3B) to which the result of the search is to be forwarded. If so, processing continues with step 610. If not, processing  
15 continues with step 612.

At step 610, if the lookup unit is the master lookup unit 150a, the next mapper 206b-d (Fig. 3B) in the master lookup unit 150a, in the lookup unit matrix 100 is searched with the result of the search in the previous mapper 206a-c (Fig. 3B) and the respective next 8-bits of the mapper key 110b, 110c or 110d. Alternately, if the lookup  
20 unit is the non-master lookup unit 150b, lookup unit 150b in the lookup unit matrix 100 is searched with the result of the search in the previous mapper 206b-c (Fig. 3B) and the respective next 8-bits of the mapper key 110c or 110d. Processing continues with step 606.

At step 612, the result of the multi-level search in the respective lookup unit  
25 150a, 150b (Fig. 3B) is forwarded from the lookup unit matrix 100 (Fig. 2) to the forwarding engine 108 (Fig. 2) as the route index 102 (Fig. 2). Processing returns to step 600 to wait to receive another mapper key 104a, 104b (Fig. 2) from the forwarding engine 108 (Fig. 2).

At step 614, in a subsequent search for mapper key 104b in non-master lookup unit 150b (Fig. 2), a search is performed for the next 24-bits of the search key 104 in the second mapper 206b (Fig. 3B) with the first 8-bits of the mapper key 110b and the search result 106 (Fig. 3B) from the search of master lookup unit 150a. Processing  
5 continues with step 606 to examine the result of the search in mapper 206b.

The lookup unit matrix can provide route indexes corresponding to a search key that is longer than the lookup unit's mapper key by performing searches of a plurality of lookup units in the lookup unit matrix 100. The same lookup unit matrix can provide a route index for an IPv4 address in a single search cycle of a master lookup unit 150a  
10 and a route index for an IPv6 address in a search of lookup unit matrix 100 including a plurality of lookup units 150a, 150b .

A lookup unit can be used to store routes for IPv4 address which are searchable in a single search cycle of the lookup unit. A plurality of lookup units can be combined in a lookup unit matrix to store routes for search keys longer than the lookup unit's  
15 mapper key.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.